REMARKS

Claims 23-36 are pending in the above-referenced application. Claims 23 and 30 are independent claims.

Applicant amended claim 23 to correct a misspelling. No new matter was added.

The examiner objected to claim 27 as being duplicative.

Applicant studied claim 27 in relation to claims 23-26, 28, and 29 and found no duplication.

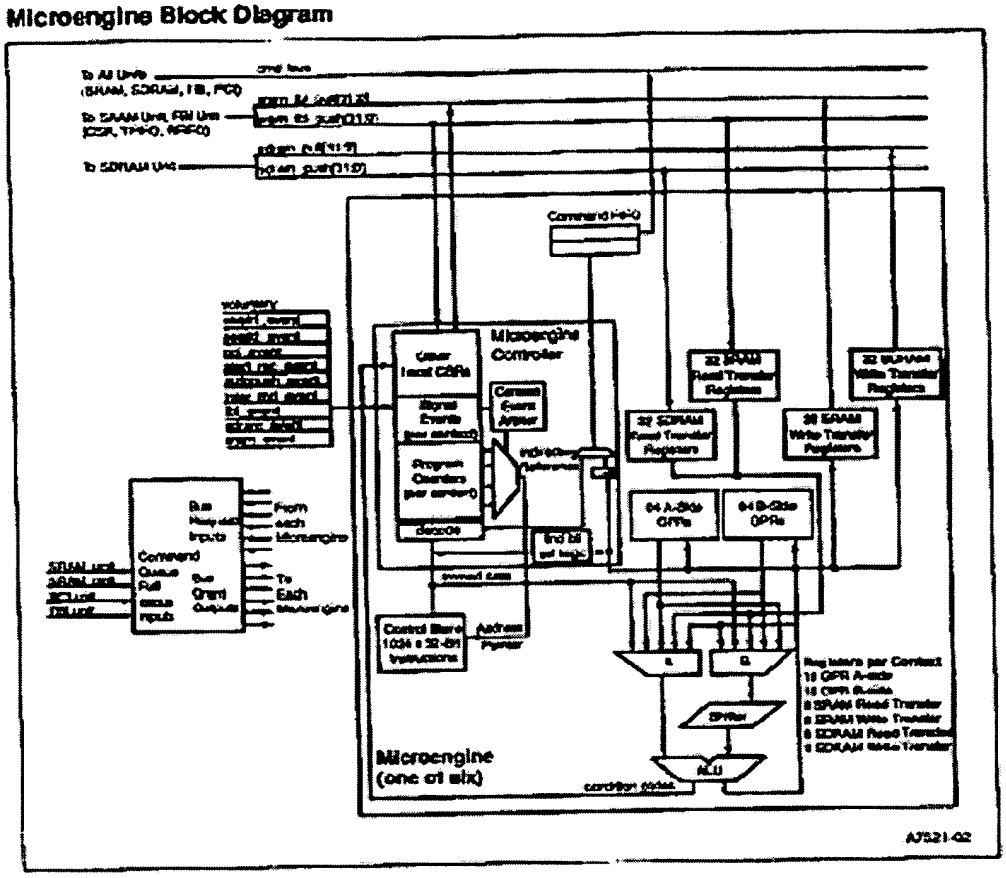
The examiner uses the June 2000 ISX1200 hardware to reject claims 23-36 as being anticipated.

Applicant disagrees. Claims 23 and 30, as amended, recite "in a parallel processor comprising a controlling processor linked to a remote console system and a plurality of micro engines," or similar language. The June 2000 IXP1200 manual neither describes nor suggests a controlling processor linked to a remote console system and a plurality of micro engines.

The examiner argues that the quoted claim feature is disclosed in the June IXP1200 manual at page 4-2 and Figure 4-1.

For the convenience of the examiner, Figure 4-1 is reproduced below.

Figure 4-1. Microengine Block Diagram



No where in Figure 4-1 does the June IXP1200 manual describe or suggest a controlling processor linked to a remote console system and a plurality of micro engines.

Claims 23 and 30, as amended, recite "loading a series of hop instructions from a debug library," or similar language. The June 2000 IXP1200 manual neither describes nor suggests this quoted claim feature.

The examiner argues that the quoted claim feature is disclosed in the June 2000 IXP1200 manual at "page 4-47, section 4.17, all five bullets."

For the convenience of the examiner, section 4.17 of the June 2000 IXP 1200 manual is reproduced below.

4.17 Debugging Support

Each Microengine provides hardware support for debugging software running on the IXP1200 Network Processor. Intel provides software that takes advantage of these debugging capabilities. This software includes the IXP Developer Workbench and debug libraries that execute on the StrongARM Core. The Debug libraries are written in the C language, allowing it to be ported to different operating systems.

The sections that follow describe the hardware support provided by the IXP1200 and how software can take advantage of this support to provide the following basic debugging capabilities:

- Determining if a Microengine is executing.
- Stopping, starting, and hopping the Microengines.
- Setting breakpoints.
- Reading the local register set.
- Creating a journal.

No where in section 4.17 does the June 2000 IXP1200 manual describe or suggest "loading a series of hop instructions from a debug library."

The June 2000 IXP 1200 manual merely discloses debug libraries that include five specific functions. More specifically, the debug libraries include the five specific functions reproduced above in section 4.17.

The function used to determine if a microengine is executing is disclosed in section 4.17.1, reproduced below.

4.17.1 Determining If a Microengine is Executing

The ACTIVE_CTX_STS (Active Context Status) local CSR indicates whether or not a context is currently executing on the Microengine. If so, it provides the context number and the PC of the instruction it is currently executing. The StrongARM Core can poll this register to determine if the context is running or whether it is idle.

If the Microengine context is idle, the StrongARM Core can read the CTX_n_SIG_EVENTS and CTX_n_WAKEUP_EVENTS registers for each context to determine the signal event for which the current context is waiting and which signal events have occurred.

If none of the Microengine contexts have received a signal event, the StrongARM Core can determine the next instruction that should be executed by reading the current PC for each of the contexts from the CTX_n_STS local CSR.

No loading a series of hop instructions is disclosed or suggested.

The function used to start, start and hop the microengines is disclosed in section 4.17.2, reproduced below, wherein the examiner has highlighted some text.

4.17.2 Stopping, Starting, and Hopping the Microengines

The StrongARM Core can place the Microengines into the Running and Paused states, described in Section 4.6. This allows the Microengine to execute for a period of time and then be placed in a paused state. Once a Microengine is in a paused state, information about the present state of the system (GPRs, Transfer Registers, memory, and CSRs) can be read by the StrongARM Core and transmitted to an I/O debug port (e.g., the serial port, Ethernet interface, etc.).

The user can manually place the Microengines into a paused state or a Microengine can place itself into the paused state automatically each time an instruction that causes a context switch is executed.

Since a Microengine can place itself into the paused state automatically, it is possible for the user to "hop" the execution of the Microengines from one context change point to the next. This allows the user to monitor the progress of a Microengine program as it executes in hardware. The term "hop" is used instead of the common term "step" because the execution progress is monitored after a context switch point rather than each time an instruction is executed.

No loading a series of hop instructions is disclosed or suggested. On the contrary, this function merely allows the placement of a microengines into running and paused states. "Hopping," as defined above, is merely the alternating of "running" and "pausing" from one context change point to the next.

Accordingly, claims 23 and 30 are not anticipated by the June 2000 IXP manual.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

Please apply any charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: April 4, 2005

Kenneth F. Kozik
Reg. No. 36,572

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21060726.doc